ISSN - 2447-178X

EMPÍRICABR

COOKIT: RECIPES RESEARCH PLATAFORM

Filipe de Oliveira Ataíde[1] e Danielle Freita[2]

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte[1,2]

Felipe.ataide@escolar.ifrn.edu.br [1] e danigfreitas@gmail.com [2]

Artigo submetido em 21/12/2022, aceito em 31/12/2022 e publicado em 13/02/2023
DOI: 10.15628/empirica.2015.14559

RESUMO

Tem pessoas que possuem interesse em cozinhar seus próprios alimentos, mas não sabem o que fazer com os ingredientes que possuem. Para isso, foi idealizado o CookIt, um site de pesquisa de receita com base em ingredientes, desenvolvido sob a prática profissional do curso de Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN). Nesse trabalho, é relatado o desenvolvimento da aplicação ao decorrer das Disciplinas de Desenvolvimento de Sistemas Web, Desenvolvimento de Sistemas Distribuídos e Desenvolvimento de Sistemas Corporativos, com relação à documentação, implementação e implantação do sistema. Além disso, são definidas as linguagens, conceitos e tecnologias utilizados durante esse período.

PALAVRAS-CHAVE: receitas; desenvolvimento web; Vue ; Django.

COOKIT: RECIPES SEARCH PLATFORM

ABSTRACT

There are people who have interest in cooking their own food, but don't know what to do with the ingredients they have. For this, CookIt was conceived, a recipe research site based on ingredients, developed under the professional practice of the Tecnologia em Análise e Desenvolvimento de Sistemas (TADS) course at the Federal Institute of Education, Science and Technology of Rio Grande do Sul. North (IFRN). In this work, the development of the application is reported during the classes of Desenvolvimento de Sistemas Web, Desenvolvimento de Sistemas Distribuídos and Desenvolvimento de Sistemas Corporativos, in relation to the documentation, implementation and implementation of the system. In addition, the languages, concepts and technologies used during this period are defined.

KEYWORDS: recipes; web development; Vue ; Django.

1 INTRODUÇÃO

Algumas pessoas desejam descobrir o que é possível fazer com os ingredientes que possui em casa, encontrando dificuldades e muitas vezes não tendo disposição para comprar itens extras para ter que preparar um prato. Aproximadamente 38% dos brasileiros se consideram interessados em culinária e 25% alegam ter grande conhecimento na área, de acordo com uma pesquisa realizada com mais de 27 mil consumidores de 15 a 22 anos em 22 países (GROWTH FROM KNOWLEDGE, 2015).

O CookIt, um sistema de pesquisa de receitas com base em ingredientes, foi idealizado pensando nesse público. O site filtra resultados com base nesses ingredientes como marcadores. É possível tanto filtrar como excluir resultados com base nos itens desejados e não desejados. Também conta com uma pesquisa por texto tradicional e um filtro avançado, onde é possível encontrar o que se procura com base em sabor, porções e nível de dificuldade. A Figura 1 representa o logotipo desenvolvido para o projeto.

Figura 1: Logotipo do CookIt.

Fonte: Elaboração própria.

O projeto também é pensado no desperdício de alimentos. Há uma estimativa de 17% dos alimentos disponíveis na mesa acabam no lixo (PROGRAMA DAS NAÇÕES UNIDAS PARA O AMBIENTE, 2021), e são associados com 8-10% das emissões globais de gases com efeito estufa (MBOW, ROSENZWEIG, *et al.*, 2019).

Além disso, ele é idealizado como uma rede social, onde é possível o diálogo entre diferentes usuários. Conta com as funções de publicar seus próprios pratos, além de editar e excluí-os adicionar outros aos favoritos, ou denunciá-los caso encontre algum conteúdo impróprio ou que viole as regras do site.

O projeto encontra-se na mesma categoria que sites como TudoGostoso e Cybercook, porém se diferencia destes exemplos com uma busca por filtro mais precisa; a interação entre usuários e a ausência de limitações na busca causadas por acesso pago. Também foi pensado a questão de menos propagandas por página com relação a estes sites, porém, devido ao foco no desenvolvimento da aplicação, não foram feitos estudos de marketing.

1.1 OBJETIVO GERAL

Desenvolver um sistema de busca de receitas por filtro de ingredientes dentro dos períodos letivos da prática profissional.

1.2 OBJETIVOS ESPECÍFICOS

- Identificar e avaliar os requisitos necessários para o funcionamento do sistema;

- Elaborar a documentação do projeto;
- Implantar o sistema em um servidor para visualização pública;
- Testar o sistema em busca de erros;

2 DESENVOLVIMENTO

O CookIt foi idealizado dentro de uma prática profissional, na matéria de Seminário de Desenvolvimento de Sistemas Web, no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, por meio de uma apresentação de ideias, onde cada aluno apresentou dentro de 1 (um) minuto uma ideia que consistia em uma problemática seguida de uma solução, essa sendo um sistema de software a ser desenvolvido durante três semestres letivos.

2.1 METODOLOGIA

O desenvolvimento do projeto CookIt foi realizado de acordo com a Estrutura Analítica do Projeto (EAP), detalhado na seção 3.1 e presente para visualização no Apêndice A. Os requisitos do sistema são levantados e analisados com a ajuda da Linguagem de Modelagem Unificada (UML), com um objetivo de desenhar um esqueleto da aplicação com todas as suas funcionalidades. Durante essa fase, também é feito um protótipo de Interface de Usuário (UI), para se ter uma ideia de como o sistema será visualmente.

Na fase da implementação, as tarefas foram divididas em *Sprints*, com base no método ágil *Scrum*. A primeira versão foi feita usando o Django Framework, com banco de dados SQLite. Depois, o framework foi usado para gerenciar as APIs a serem consumidas pela interface gráfica adotada pelo Vue.js. Em seguida, o site foi implantado com a ajuda de uma máquina virtual do Microsoft Azure e contêineres da ferramenta Docker.

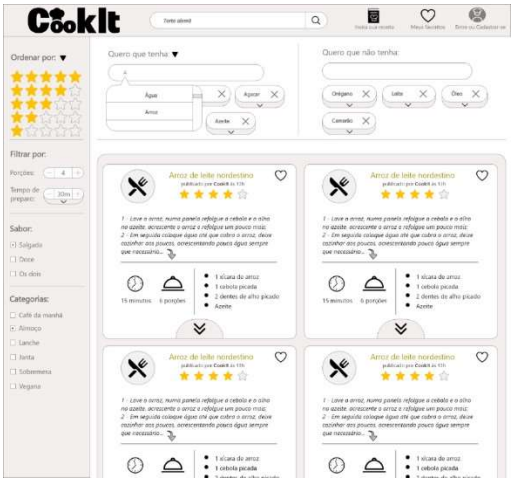
2.2 PROTOTIPAÇÃO

O protótipo inicial da interface foi desenvolvido utilizando a ferramenta Adobe XD, e inclui a tela principal, a tela de cadastro de usuário e de receita, autenticação e perfil. Ele pode ser visto na Figura 2, que mostra a tela principal, onde, no topo, a barra de navegação contendo o logotipo do site, a barra de pesquisa simples e os ícones de adicionar receita, notificações, favoritos e conta do usuário, respectivamente.

À esquerda, o filtro avançado de pesquisa, contendo ordenação, classificação por nota, quantidade de porções, tempo de preparo, sabor e categorias. No corpo do site, os campos “Quero Que Tenha” e “Quero Que Não Tenha”, que representam, em ordem, a inclusão e exclusão de receitas com base nos ingredientes acrescentados. Logo abaixo, as receitas a serem filtradas, apresentadas em forma de grade.

Figura 2: Protótipo de média fidelidade da página principal.

ISSN - 2447-178X



Fonte: Elaboração Própria.

Para refazer a interface da aplicação, foi usado o editor gráfico Figma que, assim como o Adobe XD, é usado para projetar prototipagem de Interface de Usuário e Experiência do Usuário. A mudança surgiu devido à nova escolha trazer uma liberdade maior de desenvolvimento para a equipe e de trabalho em grupo, visto que o Figma oferece essa funcionalidade gratuitamente e o Adobe XD não.

2.3 IMPLEMENTAÇÃO

A etapa de implementação marca o desenvolvimento do projeto com base na documentação desenvolvida. É acompanhada, respectivamente, nas matérias de Desenvolvimento de Sistemas Web (DSW) e Desenvolvimento de Sistemas Desenvolvidos (DSD).

A primeira versão de programação do projeto, desenvolvida em DSW, utiliza da ferramenta Django Framework tanto como *front-end* como para o *back-end*, com base nas exigências da prática profissional na fase de desenvolvimento web. Para gerenciar o banco de dados, foi escolhido o SQLite, devido à sua simplicidade. O código foi armazenado na plataforma de repositórios Gitlab, também requisito da disciplina acadêmica.

Foram criadas duas pastas dentro do projeto para armazenar os dados referentes ao usuário e às receitas, respectivamente. Na pasta de usuário, foram definidos o banco de dados e o formulário referente à autenticação de contas, enquanto em receita, o mesmo foi feito para os pratos a serem publicados, além dos ingredientes e suas avaliações. Além disso, foi feita uma pasta "*core*" para armazenar o conteúdo HTML, CSS e Javascript.

No Quadro 1, é exibido como exemplo a classe de Ingrediente. Dentro da declaração da classe, são definidas as variáveis que farão parte desta tabela no banco de dados. O campo receita, por exemplo, é uma chave estrangeira do modelo de Receita. Mais abaixo, é definido como a classe será chamada na interface do administrador, e seu nome em plural.

Quadro 1: Código da classe de Ingrediente do sistema.

ISSN - 2447-178X


```python
class Ingrediente(models.Model):
    receita = models.ForeignKey(
        Receita, on_delete=models.DO_NOTHING, blank=True, null=True)
    nome_ingrediente = models.CharField(
        'Nome Ingrediente:', max_length=100, blank=False)
    unidadeMedida = models.CharField(
        default='U',
        max_length=3,
        choices=(
            ('U', 'Unidade'),
            ('X', 'Xícara'),
            ('C', 'Colher de Sopa'),
            ('CH', 'Colher de Chá'),
            ('D', 'Dente de Alho'),
            ('M', 'Mililitro(ml)'),
            ('L', 'Litros'),
            ('G', 'Gramas(g)'),
            ('KG', 'Quilograma(kg)'),
            ('AGS', 'ao gosto'),
        )
    )
    quantidade = models.PositiveIntegerField(
        'Quantidade:', default=0, blank=False)
    def __str__(self):
        return self.nome_ingrediente
    class Meta:
        verbose_name_plural = 'Ingredientes'
```

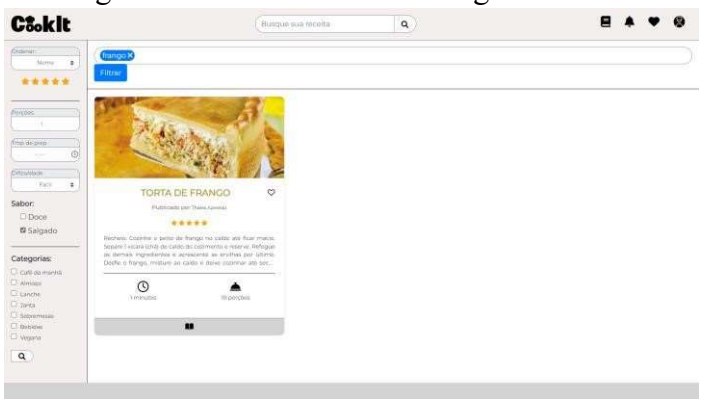
Fonte: Elaboração Própria.

Nessa versão, foi implementada a interface das páginas principal e de visualização de receita, com base no protótipo desenvolvido no Adobe XD, usando código CSS puro e, a fim de facilitar na estruturação das páginas, do auxílio do *framework* Bootstrap.

Em relação às funcionalidades do sistema, a busca simples funcionava como esperado, enquanto na busca por filtro foram implementados os campos “Quero que tenha”, porções, sabor e nível de dificuldade, deixando os campos de classificação, avaliação, tempo de preparo, porções e categorias para o período letivo seguinte.

A autenticação de usuário foi implementada, permitindo-se criar uma conta e entrar nela, porém não foi possível implementar a relação entre as contas e as receitas, deixando o campo de dono apenas como texto. Apesar disso, a criação de receitas por meio do website é realizada.

Figura 3: Primeira versão da Página inicial.



Fonte: Elaboração Própria.

A segunda versão, desenvolvida em DSD, da aplicação ocorre a divisão do *back-end* e o *front-end* da aplicação em sistemas diferentes, a fim de diminuir e identificar melhor os riscos.

O banco de dados passou a ser armazenado pela biblioteca PostgreSQL, devido à sua capacidade de suportar sistemas mais complexos.

O Django Framework continuou a trabalhar com o *back-end* da aplicação, devido à familiaridade com a ferramenta, aproveitamento de código e tempo. Para eles serem consumidos pela interface gráfica, agora separada, criou-se uma API Web utilizando a extensão Django REST Framework, visto que ela era a mais conhecida e por escolha do grupo.

No Quadro 2 é possível ver uma lista de classes do sistema seguida dos endereços usados para solicitar e enviar informações. Para enviar uma receita, por exemplo, os dados são enviados para o endereço escrito em verde na linha 2.

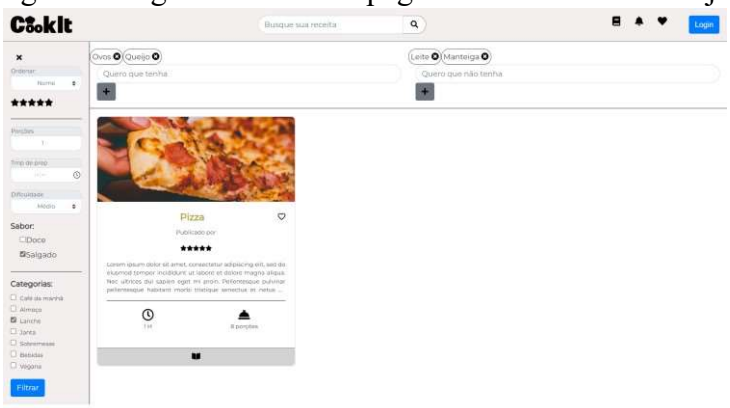
Quadro 2: Lista das classes do sistema com seus respectivos endereços de requisição.

```
1 {
2     "receita": "http://localhost:8000/api/receita/",
3     "ingrediente": "http://localhost:8000/api/ingrediente/",
4     "avaliacao": "http://localhost:8000/api/avaliacao/",
5     "user": "http://localhost:8000/api/user/",
6     "usuario": "http://localhost:8000/api/usuario/",
7     "post-receita": "http://localhost:8000/api/post-receita/"
8 }
```

Fonte: Elaboração Própria.

O Vue.js foi adotado para a interface gráfica da aplicação devido à sua popularidade e facilidade de uso. Com ele, foi possível alternar entre as páginas sem carregamento, resultando em uma navegação mais fluida e dinâmica para o usuário. A ferramenta consome o banco de dados por meio de solicitações aos endereços de requisição do Django REST Framework.

Figura 4: Segunda versão da página inicial usando Vue.js.



Fonte: Elaboração Própria.

No Quadro 3, é possível ver, das linhas 1 a 4, o método para carregar as receitas a serem mostradas na página inicial. A primeira linha mostra a requisição sendo feita pelo endereço mencionado anteriormente, enquanto na linha abaixo os dados são armazenados em um lugar onde será feita a visualização para o site.

Mais abaixo, das linhas 6 a 16, está o método responsável por filtrar as receitas com base na entrada do usuário. O endereço de requisição é mesclado com os dados do filtro enviados pelo usuário, e em seguida eles passam pelo mesmo processo que na função anterior. Se algum dos elementos do filtro for enviado vazio, a pesquisa retornará as receitas com todos os valores deste elemento, com base somente nos campos preenchidos.

Quadro 3: Código dos métodos para carregar e filtrar receitas.

```
1  async fetchReceitas({ commit }) {
2      const response = await axios.get('http://localhost:8000/api/receita/');
3      commit('setReceitas', response.data)
4  },
5
6  async basicSearch({ commit, getters }) {
7      const response = await axios.get(
8      'http://localhost:8000/api/receita/?nome_receita__icontains='
9      + getters.getPesquisa.nome_receita
10     + '&sabor_receita__in=' + getters.getPesquisa.sabor_receita
11     + '&dificuldade=' + getters.getPesquisa.dificuldade
12     + '&categoria__in=' + getters.getPesquisa.categoria
13     + '&ingredientes__nome_ingrediente__in=' + getters.getPesquisa.ingredientes
14     + '&ingredientes_not=' + getters.getPesquisa.n_ingredientes);
15     commit('setReceitas', response.data)
16 },
```

Fonte: Elaboração Própria.

Os campos Quero que tenha e Quero que não tenha foram implementados, passando apenas pela barreira de distinção entre letras maiúsculas e minúsculas. Caso o usuário digite o nome de um ingrediente começando com letra maiúscula, mas o nome registrado no sistema esteja completamente minúsculo, esse resultado não aparecerá. Este problema era ausente na primeira versão do sistema.

Na barra lateral, foram implementados os campos de sabor, dificuldade e categorias. O campo de porções não pôde ser reimplementado devido à dificuldade do grupo na nova ferramenta. Desta vez, as pesquisas aparecem em tempo real, sem a necessidade de recarregar a página, uma funcionalidade nativa do Vue.js.

A autenticação do usuário não foi implementada na nova versão devido a prazo de entrega, onde foi dada prioridade às funcionalidades principais do sistema. Isso também vale para os campos de ordenação, nota e tempo de preparo no filtro avançado. Esses itens foram deixados para a próxima etapa do desenvolvimento de sistemas.

2.4 TESTES

Um documento de plano de testes foi criado a fim de examinar o desempenho e a funcionalidade do sistema. Nele foram feitas tabelas contendo a entrada e saída esperados para cada ação realizada. Para executar os testes foi usada a ferramenta de testes do Django Framework, que funcionou tanto para a versão inicial como para a segunda versão do projeto. Os testes automatizados foram feitos com a ajuda da ferramenta Selenium, uma das ferramentas propostas pelo professor de Teste de Software.

O Quadro 4 apresenta o código do teste de busca simples. Nas linhas 1 e 2, declara-se, respectivamente, a classe e, dentro dela, o método a ser executado. Nas linhas 3 e 4, São

definidos o navegador e o endereço onde ocorrerá o teste. E nas linhas 5 e 6, ocorre a pesquisa automatizada.

No Quadro 5 é mostrado o progresso do teste sendo executado em um terminal, onde é possível ver, na linha 1, a criação de um banco de dados virtual. Na linha 2, a checagem por algum erro no código do projeto. Na linha 5, é mostrado o tempo de execução do teste. E por último, na linha 8, o banco de dados virtual é destruído.

Quadro 4: Código do teste de busca simples.

```
1 class BuscaSelTestCase(TestCase):
2     def test_busca_simples_nome_X(self):
3         self.driver=Firefox()
4         self.driver.get('http://localhost:8080/')
5         self.driver.find_element_by_name('search-input').send_keys('X')
6         self.driver.find_element_by_name('search-button').click()
```

Fonte: Elaboração Própria.

Quadro 5: O teste de busca simples sendo executado no terminal do Windows.

```
1 (venv) python manage.py
2 Creating test database for alias 'default'...
3 System check identified no issues (0 silenced).
4 ----------------------------------------------------
5 Ran 1 test in 12.567s
6
7 OK
8 Destroying test database for alias 'default'...
```

Fonte: Elaboração Própria.

Os testes foram concluídos com êxito, apresentando resultados favoráveis para o sistema, sendo os únicos erros associados à inteligência artificial usado pela ferramenta de testes. Os resultados foram documentados na matéria de Plano de Testes.

3 CONCLUSÃO

Foi desenvolvido um sistema que realiza busca por marcadores com base em ingredientes busca simples com base em título, que trazem os resultados de acordo com os dados fornecidos pelo usuário. Também estão disponíveis os filtros de dificuldade, sabor e categoria. Os objetivos foram atingidos com base no conhecimento fornecido pelos professores da prática profissional, além de conteúdos on-line buscados pelo grupo de desenvolvimento.

O Projeto de Desenvolvimento de Software serve para ter como base o que esperar do mercado de trabalho na área de análise e desenvolvimento de sistemas, fornecendo os conceitos e a prática para a elaboração de um sistema completo. Também serviu para entender como gerenciar um trabalho em equipe em um ambiente de desenvolvimento.

Entre as dificuldades enfrentadas no grupo está a implementação de alguns elementos do filtro, onde o trabalho em equipe foi necessário para se chegar a uma conclusão e concluir a função principal do sistema, e prosseguir com o projeto após a saída de um dos membros. Apesar destas ocorrências, foi possível chegar até um ponto considerável no desenvolvimento do sistema.

4 REFERÊNCIAS

ADRIAN HOLOVATY, Jacob K.-M. The Django Book. **Wayback Machine**, 02 Setembro 2016. Disponível em: https://web.archive.org/web/20160902124916/http://www.djangobook.com/en/2.0/index.html .

ALEXANDREA, Jordana. What Is Bootstrap? **Hostinger Tutorials**, março 18 2022. Disponível em: https://www.hostinger.com/tutorials/what-is-bootstrap/.

APIGEE. **Web API Design:** The Missing Link.

DJANGO REST FRAMEWORK. Home. **Django REST framework**, julho 2022. Disponível em: https://www.django-rest-framework.org/.

DOCKER DOCUMENTATION. Docker overview. **Docker Documentation**, julho 2022. Disponível em: https://docs.docker.com/get-started/overview/.

DRUMOND, Claire. O que é o scrum? **Atlassian Agile Coach**, julho 2022. Disponível em: https://www.atlassian.com/br/agile/scrum.

GROWTH FROM KNOWLEDGE. Consumers attitudes and time spent cooking. **Growth from Knowledge**, 31 março 2015. Disponível em: https://www.gfk.com/insights/consumersattitudes-and-time-spent-cooking.

GUEDES, Gilleanes T. A. **UML 2:** Uma Abordagem Prática - 3ª Edição.

INCAU, Caio. **Vue.js:** Construa aplicações incríveis.

JR., Hélio E. **Computação em nuvem com o Office 365**.

MBOW, Cheikh et al. Food security. In: SHUKLA, Priyadarshi R., et al. **Climate Change and Land:** an IPCC special report on climate change, desertification, land degradation, sustainable

land management, food security, and greenhouse gas fluxes in terrestrial ecosystems. [S.l.]: [s.n.], 2019.

MDN CONTRIBUTORS. CSS. **MDN Web Docs**, 26 junho 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS.

MDN CONTRIBUTORS. Iniciando com HTML. **MDN Web Docs**, 11 fev 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/HTML/Introduction_to_HTML/Getting_started.

MDN CONTRIBUTORS. JavaScript. **MDN Web Docs**, 2 ago 2021. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript.

MILANI, André. **PostgreSQL - Guia do Programador**.

MUNHOZ, Júlia V. Entenda o que é mobile first e conheça as suas principais vantagens. **Moblee**, julho 2022. Disponível em: https://www.moblee.com.br/blog/mobile-first-principaisvantagens/.

PROGRAMA DAS NAÇÕES UNIDAS PARA O AMBIENTE. **Food Waste Index Report 20211 (Relatório do Índice de Desperdício Alimentar 2021)**. Nairobi. 2021.

PROJECT MANAGEMENT INSTITUTE. **Project Management Body of Knowledge**.

PYTHON SOFTWARE FOUNDATION. What is Python? Executive Summary. **Python.org**, julho 2022. Disponível em: https://www.python.org/doc/essays/blurb/.

SQLITE. About SQLite. **SQLite**, julho 2022. Disponível em: https://sqlite.org/about.html.

TIME DE CUSTOMER ENGINEERS DA MICROSOFT. **Modernização de Aplicação no Microsoft Azure:** Explorando o potencial da nuvem.

VUE.JS. Introdução. **Vue.js**, 1 março 2022. Disponível em: https://vuejsbr-docsnext.netlify.app/guide/introduction.html#relacao-com-elementos-customizados.

VUETIFY. Why you should be using Vuetify. **Vuetify**, julho 2022. Disponível em: https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started.